CAPITOLO 1

Introduzione all'informatica

1.1 Introduzione

Questo testo è un invito ad apprendere una delle più giovani e interessanti tra le discipline scientifiche: l'**informatica**. Ogni giorno i giornali, le riviste e le televisioni parlano di nuovi progressi nelle tecnologie che riguardano i computer: supercomputer in grado di eseguire trilioni di operazioni al secondo, reti che trasmettono immagini e filmati in tutto il mondo, computer miniaturizzati da portare al polso come orologi. Viviamo nell'era dei computer; e questo libro vuole farvi comprendere l'informatica e apprezzare le diverse aree di ricerca e studio di cui si compone.

Una persona con un medio livello di istruzione è generalmente in grado di descrivere in modo ragionevolmente accurato gli oggetti d'interesse delle principali discipline scientifiche, anche se non li ha studiati specificamente a scuola, mentre per l'informatica il discorso cambia: la maggior parte delle persone non possiede una conoscenza intuitiva dei problemi studiati in questo campo. Per esempio, certamente sapete che la biologia è lo studio degli organismi viventi e che la chimica si occupa della struttura e composizione della materia. Tuttavia, forse non sapete di che cosa tratta l'informatica. In effetti, molte persone nutrono convinzioni errate sull'informatica, come quelle descritte di seguito.

CONVINZIONE ERRATA 1: *l'informatica è lo studio dei computer*

Questa definizione, che per alcuni appare ovvia, è in realtà errata, o quanto meno incompleta. Per esempio, alcuni tra i primi e fondamentali studi teorici di informatica risalgono al periodo compreso tra il 1920 e il 1940, ben prima che nascessero i computer (queste opere pionieristiche furono inizialmente considerate come un ramo della logica e della matematica applicata; soltanto verso la fine degli anni '50 e l'inizio degli anni '60 l'informatica fu riconosciuta come disciplina scientifica autonoma e separata). Ancora oggi esistono rami dell'informatica ben distinti dallo studio delle macchine da calcolo "reali". Nell'*informatica teorica*, per esempio, si studiano le proprietà logiche e matematiche dei problemi e delle loro soluzioni. Spesso si esaminano problemi senza utilizzare i computer, ma ricorrendo a *modelli formali* di calcolo, più facili da studiare e analizzare matematicamente. Si lavora con carta e matita, senza schede e dischetti.

Questa distinzione tra computer e informatica è espressa benissimo dagli informatici Michael R. Fellows e Ian Parberry in un articolo pubblicato sulla rivista *Computing Research News*:

> *L'informatica non si occupa dei computer più di quanto l'astronomia si occupi di telescopi, la biologia di microscopi, la chimica di provette e becher. La scienza non si occupa tanto degli strumenti, ma del modo in cui li utilizziamo e di ciò che scopriamo con essi.*[1]

CONVINZIONE ERRATA 2: *l'informatica è lo studio di come scrivere programmi per computer*

Molte persone si avvicinano per la prima volta all'informatica quando imparano a scrivere programmi in un linguaggio come C++, Scheme o Java. Questo uso quasi universale della programmazione come porta di ingresso per l'informatica può creare il malinteso secondo cui l'informatica sarebbe l'equivalente della programmazione di computer. In realtà la programmazione è estremamente importante per la nostra disciplina, poiché i ricercatori la utilizzano per studiare nuove idee e realizzare e collaudare nuove soluzioni, ma si tratta ancora di uno strumento, come il computer. Quando gli informatici progettano e analizzano un nuovo approccio per risolvere un problema, o creano nuovi modi per rappresentare informazioni, implementano le proprie idee come programmi per poterle collaudare su un computer reale. Ciò consente loro di verificare se tali idee funzionano e se consentono di avvicinarsi allo scopo. Per esempio, una delle applicazioni più comuni dei computer è la ricerca in un elenco, impiegata frequentemente in problemi di grande dimensione, come quello di cercare un nome tra circa 20 milioni di nominativi presenti nell'elenco telefonico di New York City (risolveremo questo problema nel Capitolo 2). Un metodo di ricerca più efficiente potrebbe ridurre in maniera significativa il tempo che i clienti devono attendere quando chiamano un servizio informativo sugli elenchi. Supponiamo di aver progettato ciò che riteniamo una "nuova e migliore" tecnica di ricerca. Dopo averla analizzata dal punto di vista teorico, vogliamo studiarla empiricamente scrivendo un programma che la implementi, e che possiamo eseguire su un computer per misurarne le prestazioni. Questi test dimostrerebbero sotto quali condizioni la nostra nuova tecnica è (o non è) più veloce delle procedure di ricerca attualmente in uso.

In informatica non conta semplicemente la qualità con cui è realizzato un programma, ma anche i metodi che in esso sono implementati, i servizi che fornisce e i risultati che produce. Talvolta ci si immerge a tal punto nello scrivere il codice di programma da dimenticarsi che questo è soltanto un mezzo per ottenere un fine, non un fine in sé.

CONVINZIONE ERRATA 3: *l'informatica è lo studio degli usi e delle applicazioni dei computer e del software*

Generalmente, quando non si inizia con la programmazione, ci si avvicina all'informatica grazie a un corso sull'uso di computer e applicazioni software. Tali corsi

[1] Fellows, M. R., e Parberry, I. "Getting Children Excited About Computer Science", *Computing Research News*, vol. 5, n. 1 (gennaio 1993).

riguardano tipicamente applicazioni molto diffuse come elaboratori di testi, sistemi di database, software di grafica, posta elettronica e browser web.

Queste applicazioni sono ampiamente utilizzate da professionisti di tutti i settori; tuttavia, imparare l'utilizzo di un pacchetto software sta all'informatica come la patente di guida sta all'ingegneria meccanica. Molte persone *usano* software, ma l'informatico si occupa di *specificare*, *progettare*, *realizzare* e *collaudare* il software, oltre ai computer con i quali viene eseguito.

Queste tre convinzioni errate sull'informatica non sono del tutto infondate: sono semplicemente incomplete. Computer, linguaggi di programmazione, software e applicazioni fanno in effetti parte dell'informatica, ma nessuno di essi, e nemmeno tutti messi insieme, esauriscono la ricchezza e la varietà di questa disciplina.

Finora abbiamo spiegato che cosa l'informatica *non è*. A questo punto ci chiediamo che cos'è, quali sono i concetti di base, le domande fondamentali studiate dai professionisti in questo campo. È possibile catturare in una singola definizione la portata e lo scopo di questa disciplina? Risponderemo a queste importanti domande nel prossimo paragrafo e poi nel resto del libro.

1.2 Definizione di informatica

Esistono molte definizioni di informatica, ma quella che riesce meglio a catturare la ricchezza e la portata delle idee contenute in questo ramo della scienza fu proposta per la prima volta da Norman Gibbs e Allen Tucker.[2] Secondo questa definizione, il concetto centrale dell'informatica è quello di **algoritmo**. Non è possibile capire questa disciplina senza comprendere a fondo tale concetto di importanza critica.

Definizione

L'**informatica** è lo studio degli algoritmi, che comprende:

1. le loro proprietà formali e matematiche;
2. le loro implementazioni hardware;
3. le loro implementazioni linguistiche;
4. le loro applicazioni.

La definizione di Gibbs e Tucker afferma che è compito degli informatici progettare e sviluppare algoritmi per risolvere molti problemi importanti. Il processo di progettazione comprende le seguenti attività:

- studiare il comportamento degli algoritmi per determinare se sono corretti ed efficienti (proprietà formali e matematiche);
- progettare e realizzare computer in grado di eseguire gli algoritmi (implementazioni hardware);

[2] Gibbs, N. E., e Tucker, A. B. "A Model Curriculum for a Liberal Arts Degree in Computer Science", *Comm. of the ACM*, vol. 29, n. 3 (marzo 1986).

- progettare linguaggi di programmazione e tradurre in essi gli algoritmi in modo da poterli eseguire mediante l'hardware (implementazioni linguistiche);
- identificare problemi importanti e progettare applicazioni software corrette ed efficienti in grado di risolverli (applicazioni).

Poiché è impossibile cogliere il senso di questa definizione senza sapere che cos'è un algoritmo, esaminiamo più da vicino questo termine.

Nel dizionario la parola *algoritmo* è definita come segue:

> **Algoritmo** *s.m. (dal nome del matematico persiano al-Khwarizmi), sistema di regole e procedure di calcolo ben definite che portano alla soluzione di un problema con un numero finito di operazioni.*

In termini informali, un algoritmo è una sequenza ordinata di istruzioni che risolve un problema specifico. È un elenco simile al seguente:

Passo 1: fa qualcosa
Passo 2: fa qualcosa
Passo 3: fa qualcosa
. .
. .
. .
Passo N: stop, terminato

Se utilizzate questo elenco seguendo con cura le istruzioni nell'ordine specificato, una volta giunti al termine avrete risolto il problema. Tutte le operazioni utilizzate per realizzare algoritmi rientrano in una delle seguenti tre categorie.

1. *Operazioni sequenziali.* Un'istruzione sequenziale esegue una singola attività ben definita. Terminata l'attività, l'algoritmo passa all'operazione successiva. Solitamente le operazioni sequenziali sono espresse come semplici frasi dichiarative.
   - Aggiungi un cucchiaio di burro all'impasto nella scodella.
   - Sottrai l'importo dell'assegno dal saldo del conto corrente.
   - Imposta il valore di x a 1.
2. *Operazioni condizionali.* Si tratta delle istruzioni di un algoritmo che "pongono una domanda". L'operazione successiva è selezionata sulla base della risposta fornita alla domanda.
   - Se l'impasto è troppo secco, aggiungi mezzo bicchiere d'acqua nella scodella.
   - Se l'importo dell'assegno è minore o uguale al saldo del conto corrente, allora paga l'assegno; altrimenti, informa la persona che l'assegno non è coperto.
   - Se x è diverso da 0, allora imposta y a $1/x$; altrimenti, stampa un messaggio di errore che informi dell'impossibilità di dividere per 0.
3. *Operazioni iterative.* Si tratta delle istruzioni "di ciclo" di un algoritmo. Indicano di non proseguire con l'istruzione successiva, ma di tornare indietro e ripetere l'esecuzione di un precedente blocco di istruzioni.
   - Ripeti le due operazioni precedenti finché l'impasto si è ispessito.
   - Finché vi sono ancora assegni da elaborare, esegui i seguenti cinque passaggi.
   - Ripeti i passaggi 1, 2 e 3 finché il valore di y è uguale a +1.

Utilizziamo gli algoritmi (anche se non li chiamiamo così) ogni volta che seguiamo un elenco di istruzioni per montare un giocattolo, cucinare un dolce, calcolare il saldo di un conto o iscrivere un ragazzo a scuola. Un buon esempio di algoritmo utilizzato nella vita quotidiana è fornito dalla serie di istruzioni riportata nella Figura 1.1, per programmare un videoregistratore in modo da registrare una sequenza di spettacoli televisivi. Notate i tre tipi di istruzioni presenti in questo algoritmo: sequenziali (passaggi 2, 4, 5, 6 e 8), condizionali (passaggi 1 e 7), iterative (passaggio 3).

Figura 1.1
Un esempio di algoritmo: programmazione del videoregistratore.

Algoritmo per programmare il videoregistratore

Passo 1	Se l'orologio e il calendario non sono regolati correttamente, passa alla pagina 9 del manuale d'uso e segui le istruzioni fornite, prima di procedere con il Passo 2.
Passo 2	Inserisci una videocassetta vuota nel videoregistratore.
Passo 3	Ripeti i passi da 4 a 7 per ciascun programma che vuoi registrare.
Passo 4	Inserisci il numero del canale che vuoi registrare e premi il pulsante CANALE.
Passo 5	Inserisci l'ora a cui vuoi iniziare la registrazione e premi il pulsante ORA-INIZIO.
Passo 6	Inserisci l'ora a cui vuoi interrompere la registrazione e premi il pulsante ORA-FINE. Questo completa la registrazione di un programma.
Passo 7	Se non vuoi registrare nient'altro, premi il pulsante FINE-PROG.
Passo 8	Spegni il videoregistratore. Ora l'apparecchio è in modalità TIMER, pronto a registrare.

I matematici utilizzano continuamente algoritmi; gran parte del lavoro svolto dai primi scienziati greci, romani, persiani e indiani comportava la scoperta di algoritmi per importanti problemi della geometria e dell'aritmetica, come l'*algoritmo di Euclide* per trovare il massimo divisore comune di due numeri interi positivi (questo algoritmo vecchio di 2300 anni è riportato nell'Esercizio 7 al termine del capitolo). Abbiamo studiato vari algoritmi perfino alle scuole elementari, anche se non lo sapevamo; per esempio, in prima elementare abbiamo imparato un algoritmo per sommare due numeri come

$$\begin{array}{r} 47 \\ +\ 25 \\ \hline 72 \end{array}$$

Le istruzioni fornite dall'insegnante erano queste: prima somma la colonna di numeri più a destra (7 + 5), ottenendo il valore 12. Scrivi il 2 sotto la linea e riporta l'1 nella colonna a sinistra. Ora passa alla colonna a sinistra sommando (4 + 2) e il valore riportato di 1, ottenendo 7. Scrivi questo valore sotto la linea, così ottieni la soluzione: 72.

Abu Ja'far Muhammad ibn Musa al-Khwarizmi (780-850)

La parola *algoritmo* deriva dal cognome di Muhammad ibn Musa al-Khwarizmi, un famoso matematico persiano vissuto tra il nono e il decimo secolo. A lui si deve anche uno dei primi testi scolastici di matematica, dal cui titolo *Kitab al jabr w'al muqabala* (che in italiano si può tradurre approssimativamente in "regole di ripristino e riduzione") deriva la parola algebra (al-jabr in arabo significa "riduzione"). Il cognome al-Khwarizmi fu latinizzato in *algoritmi*.

Questo algoritmo, che abbiamo imparato da bambini, può essere scritto in modo formale come una sequenza esplicita di istruzioni (Figura 1.2). Anche qui notate i tre tipi di istruzioni utilizzate per realizzare l'algoritmo: sequenziali (passi 1, 2, 4, 6, 7, 8 e 9), condizionali (passo 5) e iterative (passo 3).

Figura 1.2
Algoritmo per sommare due numeri di m cifre.

Algoritmo per sommare due numeri di *m* cifre

Dati: $m \geq 1$ e due numeri positivi ognuno contenente m cifre, $a_{m-1}\ a_{m-2}, \ldots, a_0$ e $b_{m-1}\ b_{m-2}, \ldots, b_0$

Trovare: $c_m c_{m-1}\ c_{m-2}, \ldots, c_0$, dove $c_m c_{m-1}\ c_{m-2}, \ldots, c_0 = (a_{m-1}\ a_{m-2}, \ldots, a_0) + (b_{m-1}\ b_{m-2}, \ldots, b_0)$

Algoritmo:

Passo 1 Imposta il valore di *riporto* a 0.

Passo 2 Imposta il valore di i a 0.

Passo 3 Finché il valore di i è minore o uguale a $m - 1$, ripeti le istruzioni dei passi da 4 a 6.

Passo 4 Somma le due cifre a_i e b_i al valore corrente di *riporto* per ottenere c_i.

Passo 5 Se $c_i \geq 10$, allora riporta c_i a $(c_i - 10)$ e imposta il valore di *riporto* a 1; altrimenti, imposta il nuovo valore di *riporto* a 0.

Passo 6 Somma 1 a i, spostandoti di una colonna a sinistra.

Passo 7 Imposta c_m al valore di *riporto*.

Passo 8 Stampa la soluzione finale, $c_m c_{m-1}\ c_{m-2}, \ldots, c_0$.

Passo 9 Stop.

Anche se a prima vista non si nota, questo è lo stesso "algoritmo di addizione decimale" che si impara alle scuole elementari. Se lo seguite con rigore, produce sempre il risultato corretto. Vediamolo all'opera.

Somma (47 + 25)
$m = 2$
$a_1 = 4$ $a_0 = 7$
$b_1 = 2$ $b_0 = 5$
} **Input**

Passo 1: *riporto* = 0.
Passo 2: $i = 0$.
Passo 3: ripeti i passi da 4 a 6 finché i è minore o uguale a 1.

Prima ripetizione del ciclo (i ha il valore 0)
Passo 4: Somma ($a_0 + b_0 +$ *riporto*), che è 7 + 5 + 0, perciò $c_0 = 12$.
Passo 5: Poiché $c_0 \geq 10$, reimposta $\mathbf{c_0}$ **a 2** e *riporto* a 1.
Passo 6: Reimposta i a (0 + 1) = 1. Poiché i è minore o uguale a 1, torna al passo 4.

Seconda ripetizione del ciclo (i ha il valore 1)
Passo 4: Somma ($a_1 + b_1 +$ *riporto*), che è 4 + 2 + 1, perciò $\mathbf{c_1} = 7$.
Passo 5: Poiché $c_1 < 10$, reimposta *riporto* a 0.
Passo 6: Reimposta i a (1 + 1) = 2. Poiché i è maggiore di 1, non ripetere il ciclo ma vai al passo 7.
Passo 7: Imposta $\mathbf{c_2 = 0}$.
Passo 8: Stampa il risultato $c_2\ c_1\ c_0 = 072$ (valori in **grassetto**).
Passo 9: Stop.

Abbiamo raggiunto il termine dell'algoritmo, che ha prodotto come risultato la somma dei numeri 47 e 25, ovvero il numero di tre cifre 072. Un algoritmo più elegante eliminerebbe lo zero inutile all'inizio del numero, quando l'ultimo valore di *riporto* è zero; tale modifica costituisce un esercizio al termine del capitolo. Provate a utilizzare l'algoritmo della Figura 1.2 con un'altra coppia di numeri per comprenderne a fondo il funzionamento.

L'algoritmo di addizione mostrato nella Figura 1.2 è una rappresentazione altamente formalizzata di una tecnica che la maggior parte di noi apprende in prima o seconda elementare e che praticamente tutti sanno utilizzare in modo informale. E allora perché esprimere in modo così complicato un'attività semplice come quella di sommare due numeri?

Gli algoritmi sono fondamentali nell'informatica perché:

se siamo in grado di specificare un algoritmo per risolvere un problema, allora possiamo automatizzare la risoluzione del problema.

Una volta specificato formalmente un algoritmo, possiamo costruire una macchina (o scrivere un programma, o ingaggiare una persona) che svolga i passaggi in esso contenuti. La macchina (o il programma, o la persona) non deve necessariamente comprendere i concetti o le idee alla base dell'algoritmo, ma semplicemente eseguire il passo 1, il passo 2, il passo 3... esattamente come sono scritti. Nella terminologia informatica la macchina, il robot, la persona o l'entità che esegue i passaggi dell'algoritmo si chiama **agente di calcolo**.

L'informatica può essere considerata come la scienza della risoluzione di problemi mediante algoritmi. Gran parte del lavoro di ricerca e sviluppo in questa disciplina consiste nello scoprire algoritmi efficienti e corretti per un'ampia varietà di problemi interessanti, studiare le loro proprietà, progettare linguaggi di programmazione in cui tali algoritmi possano essere codificati, pianificare e costruire computer in grado di eseguire automaticamente tali algoritmi in maniera efficiente.

A prima vista potrebbe sembrare che ogni problema possa essere risolto mediante algoritmi. Tuttavia, come vedremo nel Capitolo 11, vi è il fatto sorprendente (dimostrato per la prima volta dal logico tedesco Kurt Gödel all'inizio degli anni '30) che esistono problemi per cui non può esistere una soluzione algoritmica generale. Questi problemi sono, in un certo senso, *irrisolvibili*. Indipendentemente dal tempo e dalla fatica utilizzati dai ricercatori, nessuna soluzione sarà mai trovata. La scoperta di Gödel, che mise in subbuglio il mondo della matematica, pone effettivamente un limite alle capacità dei computer e degli informatici.

Esistono anche problemi per cui è possibile specificare un algoritmo, ma tali per cui un agente di calcolo richiederebbe talmente tanto tempo per eseguirlo, che la soluzione sarebbe sostanzialmente inutile. Per esempio, per vincere a scacchi con il computer potremmo utilizzare l'approccio detto *a forza bruta*: data una posizione sulla scacchiera come input, il computer esaminerebbe tutte le possibili mosse lecite, poi tutte le possibili contromosse che l'avversario potrebbe fare, poi tutte le possibili mosse successive e così via. Questa analisi continuerebbe fino a raggiungere la vittoria, la sconfitta o una posizione di parità. Con tali informazioni a disposizione il computer potrebbe scegliere in maniera ottimale la prossima mossa. Se, per semplicità, supponiamo che vi siano 40 mosse possibili per ciascuna posizione data sulla scacchiera, e che servano circa 30

mosse per arrivare a una conclusione, allora il numero totale di posizioni sulla scacchiera che il programma a forza bruta dovrebbe esaminare è

$$\underbrace{40 \times 40 \times 40 \times \ldots \times 40}_{30 \text{ volte}} = 40^{30}, \text{ circa } 10^{48}$$

Se potessimo costruire un computer in grado di calcolare 1 trilione (10^{12}) di posizioni sulla scacchiera per secondo (molto oltre gli attuali livelli tecnologici), esso impiegherebbe circa 30 000 000 000 000 000 000 000 000 000 anni per fare la prima mossa! Naturalmente, la tecnica della forza bruta non è adatta per giocare una reale partita a scacchi con il computer.

Esistono anche problemi che non sappiamo ancora *come* risolvere mediante algoritmi. Molti di questi riguardano attività che richiedono un certo grado di ciò che chiamiamo "intelligenza". Per esempio, dopo pochi giorni dalla nascita un neonato riconosce la faccia di sua madre tra le molte che vede. In pochi mesi inizia a sviluppare capacità di controllo motorio e sensoriale coordinato ed è in grado di pianificare come usarle: come andare dal box al giocattolo sul pavimento senza inciampare nella sedia o nel tavolo che sta in mezzo. Dopo pochi anni il bambino inizia a sviluppare potenti capacità di linguaggio e di ragionamento astratto.

Noi diamo per scontate tutte queste capacità, ma le operazioni appena citate (discernimento visivo avanzato, risoluzione di problemi di alto livello, ragionamento astratto, comprensione del linguaggio naturale) non possono essere riprodotte bene (alcune non possono esserlo affatto) utilizzando i computer e i software oggi disponibili. Il motivo principale è che i ricercatori non sanno ancora come specificare queste operazioni in modo algoritmico. In pratica non sanno come specificare formalmente una soluzione in una procedura dettagliata passo passo. Noi uomini siamo in grado di svolgere queste operazioni semplicemente utilizzando gli "algoritmi" che stanno nella nostra testa. Per apprezzare questo problema, immaginate di descrivere algoritmicamente con esattezza i passaggi che seguite quando dipingete un quadro, componete una poesia o formulate un business plan.

La risoluzione di problemi in modo algoritmico presenta molte varianti. A volte le soluzioni non esistono; a volte una soluzione è troppo inefficiente per essere utile; a volte una soluzione non è ancora nota. Tuttavia, la scoperta di una soluzione algoritmica ha conseguenze enormemente importanti: come abbiamo notato in precedenza, se possiamo creare un algoritmo corretto ed efficiente per risolvere un problema, e se lo codifichiamo in un linguaggio di programmazione, possiamo sfruttare la velocità e la potenza dei computer per automatizzare la risoluzione e produrre il risultato desiderato. Di questo si occupa l'informatica.

Le origini

Non esiste una data singola che segni la nascita dell'informatica. I primi lavori teorici sui fondamenti logici dell'informatica risalgono al 1930; i primi computer elettronici apparvero negli anni 1940-1946. La prima macchina commerciale, l'UNIVAC-I, apparve nel marzo 1951, data che segna l'inizio dell'industria dei computer. Il primo linguaggio di alto livello (basato sul linguaggio naturale) fu il FORTRAN, che alcuni fanno risalire al 1957 segnando la nascita dell'industria del software. Con la sua età compresa tra circa 40 e 70 anni a seconda di quale "evento di nascita" si consideri, rispetto alle discipline scientifiche classiche come matematica, fisica, chimica e biologia, l'informatica è il fratellino più giovane.

1.3 Algoritmi

1.3.1 Definizione formale di algoritmo

Definizione

Algoritmo: un insieme ordinato di operazioni non ambigue ed effettivamente computabili che, quando eseguito, produce un risultato e si arresta in un tempo finito.

La definizione formale di algoritmo è piuttosto stringente e contiene diverse idee importanti. La esaminiamo punto per punto, analizzandone tutte le componenti.

.... un insieme ordinato...

Un algoritmo è un insieme di operazioni, per le quali deve esistere un *ordinamento* chiaro e non ambiguo. Ciò significa che sappiamo quale operazione eseguire per prima e quale eseguire dopo averne completata una. Sempre in modo puntuale. Dopo tutto, non possiamo attenderci che un agente di calcolo esegua correttamente le nostre istruzioni se non sa bene quale eseguire.

Considerate il seguente "algoritmo" ripreso da un flacone di shampoo, che riporta le istruzioni per l'uso del prodotto.

Passo 1: Bagna i capelli
Passo 2: Insapona
Passo 3: Sciacqua
Passo 4: Ripeti

Al passo 4, quali operazioni vanno ripetute? Se torniamo al passo 1 bagneremo inutilmente i capelli (dovrebbero essere ancora umidi per il passo precedente). Se torniamo al passo 3 non puliremo meglio i capelli, perché non riutilizzeremo lo shampoo. L'istruzione Ripeti al passo 4 è ambigua perché non specifica chiaramente che cosa fare. Perciò viola il requisito di ordinamento dell'algoritmo (presenta anche un altro problema ancora più grave: non si ferma mai! Parleremo di questo problema più avanti). Istruzioni come:

- Torna indietro e riesegui (riesegui che cosa?)
- Riparti (da dove?)
- Se hai compreso questo materiale, puoi saltare avanti (fin dove?)
- Esegui la parte 1 o la parte 2 (come si decide quale eseguire?)

sono ambigue e possono creare confusione e incertezza su quali operazioni eseguire. Dobbiamo essere estremamente precisi nello specificare l'ordine in cui le operazioni devono essere eseguite. Un modo per ottenere ciò consiste nel numerare i passi dell'algoritmo e utilizzare tali numeri per specificare il corretto ordine di esecuzione. Per esempio, le operazioni ambigue riportate in precedenza potrebbero essere rese più precise nel modo seguente:

- Torna al passo 3 e continua l'esecuzione da lì.
- Riparti dal passo 1.
- Se hai compreso questo materiale, salta alla riga 21.

- Se hai 18 anni o più, esegui la parte 1 cominciando dal passo 9; altrimenti, esegui la parte 2 cominciando dal passo 40.

... non ambigue ed effettivamente computabili...

Gli algoritmi sono costituiti da elementi detti "operazioni", ma che cosa sono queste ultime? Che tipi di "mattoncini" si possono utilizzare per costruire un algoritmo? La risposta è che le operazioni utilizzate in un algoritmo devono soddisfare due criteri: essere *non ambigue* ed essere *effettivamente computabili*.

Ecco un possibile "algoritmo" per cucinare una torta alle ciliege:

Passo 1: Prepara la base
Passo 2: Prepara il ripieno alle ciliege
Passo 3: Metti il ripieno nella base
Passo 4: Cuoci nel forno a 200 °C per 45 minuti

Per un pasticcere professionale, questo algoritmo andrebbe bene. Il professionista saprebbe bene come eseguire ciascuna delle operazioni elencate. Un novizio, invece, probabilmente non avrebbe problemi con i passi 3 e 4, ma troverebbe difficoltà con i passi 1 e 2, e chiederebbe aiuto. Sarebbe meglio, quindi, fornire istruzioni più dettagliate.

Passo 1: Prepara la base
  1.1 Prendi tre tazze di farina
  1.2 Passa al setaccio la farina
  1.3 Mescola la farina setacciata con un etto di burro e mezza tazza d'acqua
  1.4 Impasta due dischi da 25 cm di diametro
Passo 2: Prepara il ripieno alle ciliege
  2.1 Apri un vasetto di marmellata di ciliege da 400 grammi e versa il contenuto in una ciotola
  2.2 Aggiungi un pizzico di cannella e noce moscata, e mescola

Con queste informazioni aggiuntive la maggior parte di noi, anche i poco esperti di cucina, capirebbe che cosa fare e sarebbe in grado di eseguire questo algoritmo di pasticceria. Tuttavia, forse dei bambini potrebbero avere ancora delle difficoltà con alcune operazioni. Per loro proseguiamo nel processo di semplificazione e descriviamo i passi ambigui in termini ancora più elementari. Per esempio, l'agente di calcolo che esegue l'algoritmo potrebbe non conoscere il significato dell'istruzione "Passa al setaccio la farina" al passo 1.2, perciò spieghiamola meglio.

  1.2 Passa al setaccio la farina
    1.2.1 Prendi il setaccio, che è l'oggetto mostrato a pagina 9 del tuo libro di cucina, e posizionalo direttamente sopra una ciotola da un litro
    1.2.2 Versa la farina sopra il setaccio e gira la manovella in senso antiorario
    1.2.3 Lascia che tutta la farina passi attraverso il setaccio nella ciotola

Ora anche un bambino dovrebbe essere in grado di eseguire le operazioni indicate. Ma se non fosse così, dovremmo semplificare ancora di più finché ogni operazione, ogni frase, ogni parola risultasse compresa senza margini di dubbio.

Un'operazione è **non ambigua** quando può essere compresa ed eseguita direttamente dall'agente di calcolo senza necessità di ulteriori semplificazioni o spiegazioni.

Quando un'operazione è non ambigua, è detta **operazione primitiva** o semplicemente **primitiva** dell'agente di calcolo che esegue l'algoritmo. Un algoritmo deve essere composto soltanto da primitive. Naturalmente le operazioni primitive di individui (o macchine) diversi variano in base alla complessità, all'esperienza e all'intelligenza dell'individuo, come abbiamo visto per la ricetta della torta alle ciliege, che varia in base all'esperienza di pasticceria della persona che esegue le istruzioni. Quindi, un algoritmo adatto per un agente di calcolo potrebbe non esserlo per un altro agente.

Una delle domande più importanti a cui rispondiamo in questo libro è: "Quali sono le operazioni primitive di un tipico computer moderno?". Quali operazioni possono essere "comprese" da un processore hardware, nel senso che possono essere eseguite direttamente, e quali altre devono essere ulteriormente raffinate e semplificate?

Tuttavia, non è sufficiente che un'operazione sia comprensibile. Deve anche essere *eseguibile* dall'agente di calcolo. Se un algoritmo mi dice di sbattere le braccia velocemente e volare, capisco perfettamente che cosa mi indica di fare, ma non sono in grado di farlo. "Eseguibile" significa che deve esistere un processo computazionale che consenta all'agente di calcolo di completare l'operazione con successo. Il termine formale utilizzato per indicare ciò è **effettivamente computabile**.

Per esempio, riportiamo di seguito una tecnica non corretta per trovare e stampare il 100-esimo numero primo (un numero primo è un numero divisibile soltanto per 1 e per sé stesso, come 2, 3, 5, 7, 11, 13, ...).

Passo 1: Genera un elenco L di tutti i numeri primi: L_1, L_2, L_3, ...
Passo 2: Ordina l'elenco L in ordine crescente
Passo 3: Stampa il 100-esimo elemento dell'elenco, L_{100}
Passo 4: Stop

Il problema di queste istruzioni sta al passo 1, "Genera un elenco L di *tutti* i numeri primi...". Tale operazione non può essere portata a termine, perché i numeri primi sono infiniti, perciò non è possibile generare l'elenco desiderato in un tempo finito. Non esiste un processo computazionale in grado di farlo, perciò l'operazione descritta al passo 1 non è effettivamente computabile.

Ecco altri esempi di operazioni non effettivamente computabili:

- Scrivi il valore decimale esatto di π (π non può essere rappresentato con assoluta esattezza).
- Imposta *media* a (*somma* ÷ *numero*) (se *numero* = 0, la divisione è indefinita).
- Imposta il valore di *risultato* a $\sqrt{N}$ (se $N < 0$, *risultato* è indefinito se si utilizzano numeri reali).
- Somma 1 al valore attuale di x (e se x attualmente non ha valore?).

Quest'ultimo esempio spiega perché abbiamo dovuto inizializzare a 0 il valore della variabile *riporto* al passo 1 della Figura 1.2. Al passo 4 l'algoritmo dice: "Somma le due cifre a_i e b_i al valore corrente di *riporto* per ottenere c_i". Se *riporto* non ha valore corrente, quando l'agente di calcolo tenta di eseguire l'istruzione al passo 4 non sa che cosa fare, perciò l'operazione non è effettivamente computabile.

... che produce un risultato...

Gli algoritmi risolvono problemi. Perché sia possibile determinare se una soluzione è corretta, l'algoritmo deve produrre un risultato osservabile da un utente, come un valore numerico, un nuovo oggetto o una modifica nell'ambiente. Senza un risultato osservabile, non saremmo in grado di dire se l'algoritmo è giusto o sbagliato. Nel caso dell'algoritmo per il videoregistratore (Figura 1.1) il risultato è una videocassetta contenente programmi televisivi registrati. L'algoritmo di addizione (Figura 1.2) produce una somma a $m + 1$ cifre.

Notate che utilizziamo la parola *risultato* invece di *risposta*. Talvolta un algoritmo non può fornire la risposta corretta, perché per un dato input, tale risposta non esiste. In questi casi l'algoritmo può produrre qualcos'altro, come un messaggio di errore, una luce rossa di avvertimento, o un'approssimazione della risposta corretta. Messaggi di errore, luci di avvertimento e approssimazioni, anche se non sono necessariamente ciò che cerchiamo, costituiscono tutti risultati osservabili.

... e termina in un tempo finito.

Un'altra importante caratteristica degli algoritmi è che devono produrre il risultato dopo l'esecuzione di un numero finito di operazioni; dobbiamo garantire che l'algoritmo alla fine raggiungerà un'istruzione del tipo: "Stop, ho finito". Abbiamo già sottolineato che l'algoritmo dello shampoo non è ben ordinato, perché non è possibile sapere quali istruzioni ripetere al passo 4. Tuttavia, anche se fosse possibile determinare quale blocco di istruzioni ripetere, l'algoritmo sarebbe comunque errato perché non termina. Viene eseguito per sempre, o finché non termina l'acqua, lo shampoo o la nostra pazienza. Questa condizione è nota come *ciclo infinito* e rappresenta un errore comune nella progettazione degli algoritmi.

Nella Figura 1.3(a) è mostrata una soluzione algoritmica al problema dello shampoo, che soddisfa tutti i criteri degli algoritmi, se supponiamo di voler lavare i capelli due volte. Tale algoritmo è ben ordinato. Ogni passo è numerato e l'esecuzione procede in sequenza, cominciando dal passo 1 e procedendo dall'istruzione i all'istruzione $i + 1$ a meno che non sia specificato altrimenti (per esempio, l'istruzione iterativa al passo 3 dice che, dopo aver completato il passo 6, occorre tornare ancora al passo 4 finché il valore di *ContaLavaggi* è 2). Ciascuna operazione è (supponiamo) chiara, non ambigua ed eseguibile dalla persona che si lava i capelli. Alla fine, l'algoritmo si arresta. Ciò è confermato dall'osservazione che *ContaLavaggi* è impostato inizialmente a 0 al passo 2; il passo 6 dice di aggiungere 1 a *ContaLavaggi* ogni volta che si insaponano e sciacquano i capelli, perciò tale variabile assumerà i valori 0, 1, 2, ... Tuttavia, l'istruzione iterativa al passo 3 dice di fermare il lavaggio quando il valore di *ContaLavaggi* raggiunge 2. A quel punto l'algoritmo va al passo 7 e termina l'esecuzione con il risultato desiderato: capelli puliti (tutto ciò è corretto, ma non aspettatevi di vedere un algoritmo come questo su un flacone di shampoo!).

Come per qualsiasi ricetta o insieme di istruzioni, esiste sempre più di un modo per scrivere una soluzione corretta. Per esempio, l'algoritmo della Figura 1.3(a) può anche essere scritto nella forma della Figura 1.3(b). Entrambe queste forme sono soluzioni corrette per il problema dello shampoo (però non sono parimenti eleganti, come vedrete nell'Esercizio 6 al termine del capitolo).

Figura 1.3(a) Una soluzione corretta al problema dello shampoo.

Passo	Operazione
1	Bagna i capelli
2	Imposta il valore di *ContaLavaggi* a 0
3	Ripeti i passi da 4 a 6 finché il valore di *ContaLavaggi* è 2
4	Insapona i capelli
5	Risciacqua i capelli
6	Somma 1 al valore di *ContaLavaggi*
7	Stop, lavaggio capelli terminato

Figura 1.3(b) Un'altra soluzione corretta al problema dello shampoo.

Passo	Operazione
1	Bagna i capelli
2	Insapona i capelli
3	Risciacqua i capelli
4	Insapona i capelli
5	Risciacqua i capelli
6	Stop, lavaggio capelli terminato

1.3.2 Risoluzione algoritmica dei problemi

Le sequenze di istruzioni mostrate nelle Figure 1.1, 1.2, 1.3(a) e 1.3(b), anche se brevi e semplici, sono esempi di soluzioni algoritmiche progettate, analizzate, implementate e collaudate dagli informatici. Le operazioni visibili in tali figure possono essere codificate in un linguaggio appropriato e fornite a un agente di calcolo (come un personal computer o un robot) perché le esegua. Il dispositivo dovrebbe seguire meccanicamente le istruzioni e completare con successo il compito, senza la necessità di comprendere i processi creativi che hanno portato a scoprire la soluzione, né i principi e i concetti alla base del problema.

Il robot si limita a eseguire i passi nell'ordine specificato (un requisito per gli algoritmi), completando con successo ciascuna operazione (altro requisito) e alla fine producendo il risultato desiderato in un tempo finito (altro requisito).

Come la rivoluzione industriale del diciannovesimo secolo consentì di utilizzare le macchine per svolgere lavori fisici ripetitivi, la "rivoluzione informatica" del ventesimo e ventunesimo secolo ci ha consentito di implementare algoritmi per meccanizzare e automatizzare l'esecuzione di attività mentali ripetitive, come la somma di lunghe colonne di numeri, la ricerca di nomi in un elenco telefonico, l'ordinamento di studenti per numero di corso, il recupero di prenotazioni alberghiere o aeree da un file contenente migliaia di record di dati. Questo processo di meccanizzazione offre la prospettiva di un'enorme crescita della produttività e consente alle persone di dedicarsi alle attività che esse sanno fare molto meglio dei computer, come creare nuove idee, stabilire regole, svolgere pianificazione di alto livello e determinare il significato dei risultati prodotti da un computer. Queste operazioni rappresentano certamente un impiego molto più efficiente di quel particolarissimo agente di calcolo che è il cervello umano.

IN PRATICA

Procuratevi una copia delle istruzioni che spiegano come:

1. iscrivervi ai corsi per l'inizio del semestre;
2. utilizzare il catalogo online per consultare la biblioteca dell'università riguardo un dato argomento;
3. utilizzare la fotocopiatrice che è disponibile nel vostro dipartimento;
4. accedere al World Wide Web.

Consultate le istruzioni e stabilite se soddisfano la definizione di algoritmo fornita in questo paragrafo; in caso negativo, spiegate perché e riscrivete ciascuna serie di istruzioni in modo che costituisca un algoritmo valido. Inoltre, stabilite se ciascuna istruzione determina un'operazione sequenziale, condizionale o iterativa.

1.4 Breve storia dell'informatica

Sebbene l'informatica non riguardi semplicemente lo studio dei computer, senza dubbio tale campo si è creato e ha guadagnato popolarità come risposta diretta alla creazione e all'uso universale delle macchine per il calcolo. Nel seguito è illustrato in sintesi lo sviluppo storico dei sistemi informatici.

L'aspetto di alcune tecnologie, quali il telefono, la lampadina e il primo volo aereo, può essere ricondotto direttamente a un singolo luogo, uno specifico individuo e un punto esatto nel tempo. Come esempi si possono citare il volo di Orville e Wilbur Wright il 17 dicembre 1903 a Kitty Hawk, North Carolina, e la famosa frase: "Signor Watson, venga qui, la voglio vedere" pronunciata da Alexander Graham Bell sul primo telefono il 12 marzo 1876.

Per i computer non si può dire qualcosa di analogo. Le macchine per il calcolo non sono apparse in un luogo preciso e in un dato giorno, come le invenzioni di qualche mente geniale. I concetti che hanno portato alla progettazione dei primi computer si sono evoluti nel corso di anni, grazie ai contributi di molte persone, ciascuna responsabile della creazione e dell'estensione del lavoro di ricercatori precedenti.

1.4.1 Il periodo iniziale: fino al 1940

Una discussione sulla storia della matematica e dell'aritmetica, invece che sull'informatica, dovrebbe risalire a 3.000 anni or sono, con i primi lavori di greci, egizi, babilonesi, indiani, cinesi e persiani. Tutte queste culture avevano interesse nei campi della matematica, della logica e del calcolo numerico e vi apportarono importanti contributi. I greci, per esempio, svilupparono i campi della geometria e della logica; i babilonesi e gli egiziani svilupparono metodi numerici per il calcolo delle radici quadrate, tabelle di moltiplicazione e trigonometriche utilizzate dai primi naviganti; i matematici indiani svilupparono sia il sistema di numerazione decimale in base 10, sia il concetto dello zero; e nel nono secolo i persiani svilupparono la soluzione di problemi algoritmici.

La prima metà del diciassettesimo secolo vide numerosi sviluppi importanti correlati all'automazione e alla semplificazione del monotono lavoro di calcolo aritmetico (la motivazione sembra essere l'improvvisa espansione della ricerca scientifica durante i secoli sedicesimo e diciassettesimo nelle aree dell'astronomia, della chimica e della medicina. Questo richiese la soluzione di problemi matematici più grandi e complessi). Nel 1614 lo scozzese John Napier inventò i **logaritmi** come metodo per semplificare

difficili calcoli matematici. Gli inizi del diciassettesimo secolo furono inoltre testimoni dello sviluppo di numerosi dispositivi meccanici nuovi e assai potenti, progettati per ridurre l'onere del calcolo aritmetico. Il primo **regolo** apparve intorno al 1622. Nel 1672 il filosofo e matematico francese Blaise Pascal progettò e costruì una delle prime **calcolatrici meccaniche** (chiamata **Pascalina**) in grado di eseguire somme e sottrazioni. Un modello di questi primi dispositivi di calcolo è illustrato nella Figura 1.4.

Anche il famoso matematico tedesco Gottfried Leibnitz (il quale, insieme a Isaac Newton, fu uno degli inventori del calcolo) fu particolarmente interessato all'idea del calcolo automatico. Studiò l'opera di Pascal e di altri e nel 1674 costruì una calcolatrice meccanica chiamata **ruota di Leibnitz**, in grado non solo di eseguire somme e sottrazioni, ma anche moltiplicazioni e divisioni. Entrambe le macchine di Pascal e Leibnitz utilizzavano gli incastri di ruote e ingranaggi per memorizzare i numeri ed eseguire operazioni aritmetiche elementari. Considerando lo stato della tecnologia disponibile a Pascal, Leibnitz e agli altri nel diciassettesimo secolo, queste prime macchine calcolatrici erano vere meraviglie meccaniche.

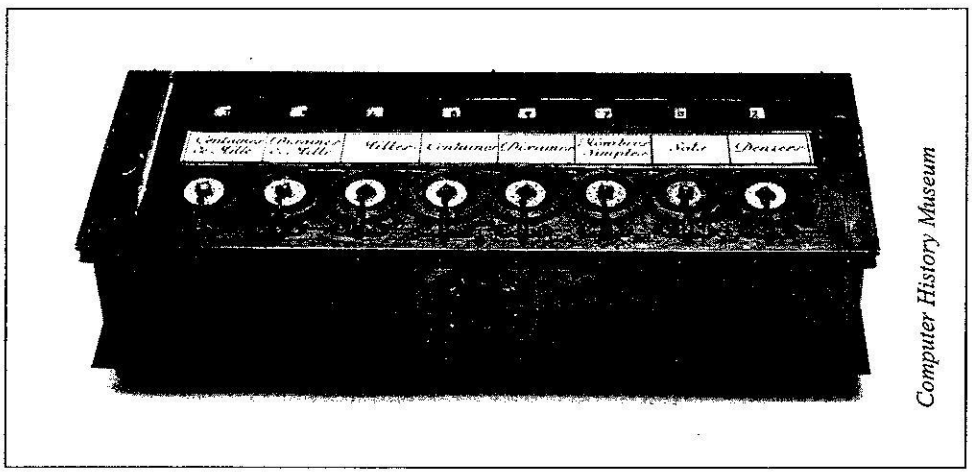


Figura 1.4
La Pascalina, una delle prime calcolatrici meccaniche.

Questi primi sviluppi nella matematica e nell'aritmetica furono importanti pietre miliari, poiché dimostrarono in che modo la meccanizzazione poteva semplificare e accelerare il calcolo numerico. La ruota di Leibnitz, per esempio, consentì ai matematici del diciassettesimo secolo di creare tabelle di funzioni matematiche assai più veloci di quanto fosse possibile calcolare a mano (nella moderna società altamente tecnologica è assai difficile credere che nel diciassettesimo secolo la creazione di una tabella di logaritmi potesse rappresentare l'impegno di un'intera vita di una persona). Tuttavia, il regolo e le macchine calcolatrici di Pascal e Leibnitz, sebbene dispositivi di alto ingegno, non erano computer. Nello specifico, erano privi di due caratteristiche fondamentali:

- non disponevano di *memoria* in cui fosse possibile archiviare le informazioni in forma leggibile dalla macchina;
- non erano *programmabili*. Era impossibile fornire *in anticipo* una sequenza di istruzioni che potessero essere eseguite dal dispositivo senza intervento manuale.

Sorprendentemente, il primo vero "dispositivo informatico" a includere entrambe queste caratteristiche non fu creato allo scopo di effettuare calcoli matematici, ma riguardò un telaio usato per la produzione di tappeti e tessuti. Fu sviluppato nel 1801 dal francese Joseph Jacquard. Egli voleva automatizzare il processo di tessitura, all'epoca attività terribilmente lenta e complicata, nella quale ciascuna riga separata della trama doveva essere predisposta della tessitrice e da un'apprendista. Per questa ragione, la maggior parte delle persone poteva permettersi soltanto lo stile di tessuto più elementare.

Jacquard progettò un telaio automatico che utilizzava **schede perforate** per creare la trama desiderata. In presenza di un foro nella scheda in una posizione particolare, un ago poteva passare attraverso la scheda, afferrare un filo dell'ordito e sollevarlo per consentire il passaggio di un secondo filo sottostante. In assenza di foro sulla scheda, l'ago non poteva passare e il filo sarebbe passato sopra l'ordito. A seconda che il filo passasse sopra o sotto l'ordito, si creava un disegno specifico. Ogni scheda perforata descriveva una riga della trama. Jacquard collegò le schede e le fece passare nel telaio, automaticamente e in sequenza, tessendo così la trama desiderata. Un disegno del **telaio di Jacquard** è illustrato nella Figura 1.5. Sulla parte superiore del dispositivo si possono vedere le file di schede perforate collegate.

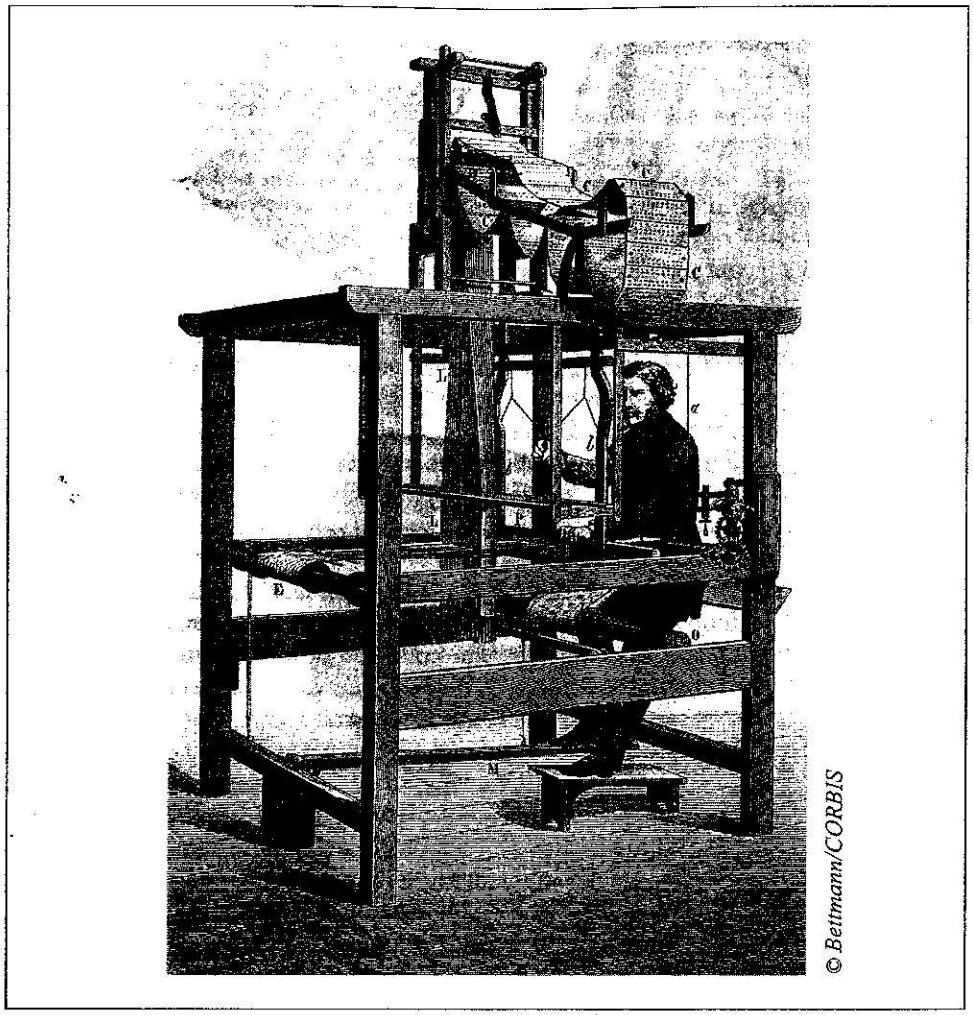
Il telaio di Jacquard rappresentò una fase importante nello sviluppo dei computer. Non solo si trattava del primo dispositivo programmabile, ma dimostrava anche come le conoscenze di un essere umano esperto (in questo caso di un mastro tessitore) potessero essere catturate in un formato leggibile dalla macchina e utilizzato per controllare un sistema che portasse a termine la stessa attività in modo automatico. Una volta creato il programma, l'esperto non era più necessario. Il più infimo apprendista poteva caricare le schede, avviare il telaio e realizzare un prodotto finito di alta qualità, più e più volte.

Questi pionieri ebbero un'enorme influenza sui progettisti e inventori che vennero dopo di loro, tra i quali un professore di matematica presso l'Università di Cambridge, Charles Babbage. Babbage era interessato al calcolo automatico; nel 1823 estese le idee di Pascal e Leibnitz e costruì un modello funzionante della più grande e sofisticata calcolatrice meccanica del suo tempo. Questa macchina, chiamata **macchina differenziale**, era in grado di eseguire somme, sottrazioni, moltiplicazioni e divisioni fino a 6 cifre significative e poteva risolvere equazioni di polinomi e anche problemi matematici complessi. Babbage tentò di costruire un modello più grande del motore differenziale, che sarebbe stato capace di operare con una precisione di 20 cifre significative, ma dopo 12 anni di lavoro dovette rinunciare al progetto. La tecnologia disponibile negli anni 1820–1830 non era così avanzata da produrre ruote e ingranaggi di estrema precisione, come richiedeva il suo progetto. Come avvenne per l'elicottero di Galileo e il sottomarino

Pubblica tecnofobia

Lo sviluppo del telaio automatizzato Jacquard e altri progressi tecnologici nell'industria tessile ebbe un tale impatto sulle potenti corporazioni attive agli inizi del diciannovesimo secolo che nel 1811 portò alla formazione del gruppo dei **Luddisti**, dal nome del loro leader Ned Ludd di Nottingham, Inghilterra. I Luddisti si opponevano violentemente a questa nuova tecnologia e si rendevano responsabili di incendi e attentati di vario tipo contro le fabbriche in cui si tentava di adottarla. Il movimento si disperse dopo pochi anni e i suoi leader furono tutti imprigionati, ma il loro nome è rimasto in uso come termine peggiorativo utilizzato per indicare un gruppo spaventato dalle novità e contrario agli ultimi sviluppi in qualunque ramo della scienza e della tecnologia, inclusa l'informatica.

Figura 1.5
Disegno del telaio di Jacquard.



atomico di Jules Verne, le idee di Babbage erano concettualmente corrette, ma in anticipo sui tempi (nel 1991 il London Museum of Science, seguendo il progetto originale di Babbage, realizzò un modello reale e funzionante della macchina differenziale. Era alto più di due metri, largo oltre tre, pesava 3 tonnellate e possedeva 4.000 parti mobili. Funzionava esattamente come previsto da Babbage).

Il matematico non smise comunque di indagare sul motore differenziale; intorno al 1830 progettò una macchina per calcolo universale assai più potente, che poteva essere configurata per risolvere una gamma ben più ampia di problemi matematici. Questa macchina aveva quattro componenti fondamentali: un **mulino** per eseguire la manipolazione aritmetica dei dati, un **deposito** per conservare i dati, un **operatore** per elaborare le istruzioni contenute nelle schede perforate e un'**unità di uscita** per collocare i risultati su schede perforate separate. Nonostante siano dovuti trascorrere quasi 110 anni prima della costruzione di un computer "reale", la macchina proposta da Babbage, chiamata **macchina analitica**, è sorprendentemente simile nel progetto a un computer moderno.

Le quattro componenti del motore analitico sono praticamente identiche nella funzione alle quattro componenti principali dei moderni sistemi informatici:

Termine di Babbage	Terminologia moderna
mulino	*unità aritmetico/logica*
deposito	*memoria*
operatore	*processore*
uscita	*ingresso/uscita*

Babbage morì prima che potesse essere realizzato un modello funzionante alimentato a vapore del suo motore analitico, tuttavia le sue idee sopravvissero, influenzando altre persone, e molti informatici moderni considerano il motore analitico il primo "vero" computer, sebbene esistesse soltanto sulla carta e nelle idee di Babbage.

Un'altra persona influenzata dal lavoro di Pascal, Jacquard e Babbage fu un giovane studioso di statistica presso l'ufficio del censimento degli Stati Uniti: Herman Hollerith. A causa del rapido aumento dell'immigrazione in America alla fine del diciannovesimo secolo, i funzionari prevedevano che, per eseguire i conteggi manualmente, sarebbero stati necessari 10-12 anni. Il censimento del 1900 sarebbe iniziato prima che fosse terminato quello precedente. Era necessario fare qualcosa.

Hollerith progettò e costruì macchine di elaborazione programmabili a schede, in grado di leggere, conteggiare e ordinare i dati immessi su schede perforate. I dati del censimento venivano codificati sulle schede mediante un dispositivo chiamato **perforatrice a tastiera**. Le schede venivano portate su un **tabulatore** per il conteggio o su un **ordinatore** per l'ordinamento alfabetico o numerico. Entrambi questi dispositivi erano programmabili (tramite fili e spine) in modo che l'utente potesse specificare quali colonne conteggiare e in quale ordine disporre le schede. Inoltre, le macchine disponevano di una piccola quantità di memoria per archiviare i risultati. In questo modo, possedevano tutte e quattro le componenti del motore analitico di Babbage.

Le macchine di Hollerith ebbero un enorme successo e furono tra i primi esempi d'uso dell'elaborazione automatica delle informazioni per la risoluzione di problemi "reali" su larga scala. Mentre il censimento del 1880 richiese 8 anni per essere comple-

Charles Babbage (1791–1871) e Ada Augusta Byron, Contessa di Lovelace (1815–1852)

Charles Babbage, figlio di un banchiere, nacque in Inghilterra nel diciottesimo secolo, in una famiglia agiata. Frequentò l'Università di Cambridge mettendo in luce un talento per matematica e scienza. Fu anche un inventore e un appassionato che amava costruire ogni sorta di apparecchi, ma il suo primo e più grande amore fu la matematica. Babbage fu enormemente impressionato dall'opera di Jacquard in Francia e passò gli ultimi 30-40 anni della propria vita cercando di costruire un dispositivo di calcolo, la macchina analitica, basata sulle idee di Jacquard.

In quella ricerca Babbage fu aiutato dalla Contessa Ada Augusta Byron, figlia del famoso peta inglese Lord Byron. La contessa fu colpita dalle idee di Babbage sulla macchina analitica, tanto da affermare: "Potremmo dire che la macchina analitica intesse motivi algebrici, come il telaio di Jacquard intesse fiori e foglie". Lady Lovelace lavorò con Babbage per specificare come organizzare le istruzioni che consentissero alla macchina analitica di risolvere un particolare problema matematico. Grazie alla sua opera pionieristica, generalmente la si considera come la prima persona nella storia a impersonare il ruolo di programmatore.

Babbage morì nel 1871 senza realizzare il suo sogno. Morì povero, perché spese per la macchina analitica tutto il suo patrimonio. La sua opera fu dimenticata fino al ventesimo secolo, quando fu fondamentale nel passaggio all'era dell'informatica.

tato, quello del 1890 terminò in soli due anni, nonostante l'incremento del 30% nella popolazione degli Stati Uniti durante tale decade.

Sebbene non si trattasse in realtà di computer universali, le macchine a schede di Hollerith furono una dimostrazione assai chiara e riuscita degli enormi vantaggi dell'elaborazione automatica delle informazioni. Questo fatto non si esaurì con Hollerith, che lasciò l'ufficio del censimento nel 1902 per fondare la società Computer Tabulating Recording Company per la produzione e vendita di tali sistemi. Egli progettò di commercializzare il suo nuovo prodotto in un Paese che stava appena entrando nell'era industriale e che, come l'ufficio del censimento, avrebbe generato ed elaborato volumi immensi di dati di inventario, produzione, contabilità e vendita. Le sue macchine a schede perforate divennero la forma dominante di apparecchiature di elaborazione dati durante la prima metà del ventesimo secolo. Nel corso di quegli anni, praticamente ogni azienda importante degli Stati Uniti disponeva di sale di elaborazione dati piene di perforatrici a tastiera, ordinatori e tabulatori, nonché di cassetti e cassetti di schede perforate. Nel 1924 la società di macchine di tabulazione di Hollerith mutò il suo nome in IBM, destinata a diventare in futuro la più grande società di informatica del mondo.

Molta strada è stata fatta dal 1640 e dalla Pascalina. Abbiamo visto lo sviluppo di calcolatrici meccaniche più potenti (Leibnitz), dispositivi di produzione automatici programmabili (Jacquard), il progetto del primo dispositivo di elaborazione (Babbage) e le applicazioni iniziali dell'elaborazione delle informazioni su larghissima scala (Hollerith). Tuttavia, non siamo ancora entrati nell'era del computer. Ciò non accadde fino a circa il 1940 e il motivo fu l'evento che, purtroppo, diede l'impulso a importanti progressi tecnologici nella storia dell'umanità: lo scoppio della guerra.

1.4.2 La nascita dei computer: 1940–1950

La seconda guerra mondiale creò un'altra serie di problemi basati sulle informazioni, di natura assai diversa. Invece di inventari, vendite e stipendi, ci si preoccupava di tabelle balistiche, dati di dispiegamento delle truppe e codici segreti. Iniziarono numerosi progetti di ricerca, finanziati in larga misura dall'esercito, per la creazione di macchine per calcolo atte a eseguire tali compiti e assistere gli Alleati nell'impegno del conflitto.

A partire dal 1931, la Marina degli Stati Uniti e IBM finanziarono congiuntamente un progetto presso l'Università di Harvard, gestito dal Professor Howard Aiken, per la costruzione di un dispositivo di calcolo chiamato Mark I. Si trattava di un computer elettromeccanico programmabile universale, che adottava una miscela di relé, magneti e ingranaggi per l'elaborazione e l'archiviazione dei dati. Il Mark I era il primo dispositivo informatico a utilizzare il sistema di numerazione binario in base 2, che illustreremo nel Capitolo 4. Utilizzava valvole e corrente elettrica per rappresentare i due valori binari: spento per lo 0, acceso per 1. Sino ad allora le macchine per calcolo avevano utilizzato la rappresentazione decimale, impiegando generalmente un ingranaggio a 10 denti, ognuno dei quali rappresentava una cifra da 0 a 9. Il Mark fu completato nel 1944, circa 110 anni dopo l'idea di Babbage del motore analitico, ed è generalmente considerato uno dei primi computer universali funzionanti. Aveva una capacità di memoria di 72 numeri e poteva essere programmato per eseguire moltiplicazioni a 23 cifre nel tempo straordinariamente breve di 4 secondi. Sebbene ciò possa far sorridere se si pensa agli

standard moderni, il Mark I fu operativo per circa 15 anni ed eseguì importanti e utili compiti matematici per conto della Marina degli Stati Uniti negli anni del conflitto.

All'incirca nello stesso periodo stava prendendo forma una macchina assai più potente, presso l'Università della Pennsylvania, in collaborazione con l'Esercito americano. Durante i primi giorni della seconda guerra mondiale, l'Esercito produceva molti nuovi pezzi di artiglieria, ma scoprì di non riuscire a produrre le tabelle di fuoco altrettanto velocemente. Tali tabelle informano il soldato su come mirare in base a elementi, quali la distanza dall'obiettivo, la temperatura, il vento e l'elevazione correnti. A causa del numero enorme di variabili e della complessità dei calcoli (che richiedono trigonometria e analisi), la realizzazione di queste tabelle di fuoco richiedeva più tempo di quella delle armi stesse.

Per risolvere il problema, l'esercito avviò nel 1943 un progetto di ricerca con J. Presper Eckert e John Mauchly dell'Università della Pennsylvania, per la creazione di un dispositivo completamente elettronico. La macchina, battezzata ENIAC (Electronic Numerical Integrator and Calculator), fu completata nel 1946 e fu il primo computer programmabile universale completamente elettronico. Questa macchina pionieristica è illustrata nella Figura 1.6.

ENIAC conteneva 18.000 valvole e riempiva quasi un interno edificio; era lungo circa 30 metri, alto 3 e pesava 30 tonnellate. A causa della natura completamente elettronica, non conteneva alcun componente meccanico lento presente nel Mark I ed eseguiva le istruzioni assai più rapidamente. L'ENIAC poteva sommare numeri a 10 cifre in circa 1/5000 di secondo e moltiplicare due numeri in 1/300 di secondo: un migliaio di volte più veloce del Mark I.

Mark I ed ENIAC sono due ben noti esempi dei primi computer, ma non sono assolutamente gli unici protagonisti di quell'epoca. Il sistema ABC (Atanasoff-Berry Computer), per esempio, progettato e costruito dal Professor John Atanasoff e dallo

Dalla collezione dell'Università della Pensylvania

Figura 1.6
Fotografia del computer ENIAC.

studente Clifford Berry presso l'Università dell'Iowa, fu in realtà il primo computer elettronico, realizzato durante il periodo 1939–1942. Tuttavia, non ricevette mai un riconoscimento equivalente, perché era utile solo per un unico compito: la risoluzione di sistemi di equazioni lineari simultanee. In Inghilterra fu realizzato nel 1943 un computer chiamato Colossus, sotto la direzione di Alan Turing, un famoso matematico e scienziato informatico di cui parleremo ancora nel Capitolo 10. Questa macchina, uno dei primi computer realizzati al di fuori degli Stati Uniti, venne impiegata per infrangere il famoso codice tedesco Enigma, ritenuto dai nazisti inviolabile. Anche Colossus non ricevette lo stesso riconoscimento di ENIAC, a causa delle segretezza che avvolgeva il progetto Enigma. La sua stessa esistenza non fu nota al pubblico generale, se non dopo molti anni dalla fine della guerra.

Pressoché nello stesso periodo in cui stava prendendo forma Colossus, un ingegnere tedesco di nome Konrad Zuse lavorava a un dispositivo informatico per l'esercito tedesco. La macchina, denominata in codice Z1, era simile concettualmente a ENIAC; un dispositivo informatico programmabile universale, completamente elettronico. Fortunatamente per le forze alleate, il progetto Z1 non fu completato prima della fine della seconda guerra mondiale.

Sebbene le macchine appena descritte (ABC, Mark I, ENIAC, Colossus e Z1) fossero computer nel vero senso della parola (disponevano di memoria ed erano programmabili), non assomigliavano ancora ai moderni sistemi informatici. Era necessario un passo ulteriore, che fu compiuto nel 1946 dall'individuo che ebbe un ruolo determinante nella creazione del computer così come lo si conosce oggi: John Von Neumann.

Von Neumann non fu soltanto uno dei più brillanti matematici di tutti i tempi, ma anche un genio in molte altre aree, tra cui fisica sperimentale, chimica, economia e informatica. Insegnava presso l'Università di Princeton e aveva lavorato con Eckert e Mauchly sul progetto ENIAC presso l'Università della Pennsylvania. Anche se tale progetto riuscì, egli riconobbe numerosi difetti in ENIAC. Nel 1946 propose un progetto di computer radicalmente diverso, basato su un modello chiamato **computer con programma memorizzato.** Fino ad allora, tutti i computer venivano programmati *esternamente* mediante fili, connettori e quadri di connessione. L'unità di memoria archiviava solo i dati e non le istruzioni. Per ciascun problema diverso, gli utenti dovevano ricablare praticamente

Dalla collezione dell'Università della Pensylvania

Inventore

Dal 1939 al 1946 molti gruppi erano impegnati nella progettazione e nella realizzazione dei primi computer, perciò non sembra possibile attribuire a una persona singola il titolo di inventore del computer digitale elettronico.

Tuttavia, non è così. Nel febbraio 1964, la Sperry Rand Corp. (ora UNISYS) ottenne un brevetto degli Stati Uniti sul computer ENIAC come primo dispositivo di calcolo interamente elettronico, con J. Presper Eckert e John Mauchly come progettisti e costruttori. Tuttavia, nel 1967 fu intentata causa contro tale brevetto. La storica sentenza della causa Honeywell contro Sperry Rand fu emessa nel 1973 (non ebbe mai adeguata copertura da parte dei media perché si era nel periodo del Watergate). Il giudice Larson annullò il brevetto ENIAC perché ritenne che Eckert e Mauchly erano stati influenzati dai precedenti studi di John Atanasoff dell'Università dello Iowa. La sentenza non fu mai appellata, perciò l'onore ufficiale di progettista e costruttore del primo computer elettronico, almeno nell'area di competenza del tribunale statunitense, va al Professor John Vincent Atanasoff.

Il 13 novembre 1990, in una cerimonia ufficiale alla Casa Bianca, il Professor Atanasoff ricevette il National Medal of Technology, da parte del Presidente George Bush, per il suo contributo pionieristico allo sviluppo del computer.

l'intero computer. I quadri di connessione dell'ENIAC, per esempio, contenevano 6.000 interruttori separati, e per riprogrammare la macchina occorreva specificare le nuove impostazioni di tutti questi interruttori, un compito assai gravoso.

Von Neumann propose che le istruzioni che controllavano il funzionamento del computer fossero codificate come valori binari e memorizzate internamente nell'unità di memoria insieme ai dati. Per risolvere un nuovo problema, invece di ricablare la macchina, occorreva riscrivere la sequenza di istruzioni, ossia creare un nuovo programma. Von Neumann inventò la programmazione così come la si conosce oggi.

Il modello di elaborazione proposto da Von Neumann includeva molte altre importanti caratteristiche che si trovano in tutti i moderni sistemi informatici e, in suo onore, questo modello di calcolo è diventato noto come **architettura di Von Neumann**. Questa architettura è illustrata in dettaglio nei Capitoli 4 e 5.

Il gruppo di ricerca di Von Neumann presso l'Università della Pennsylvania implementò le sue idee e realizzò nel 1951 uno dei primi computer con programma memorizzato, chiamato EDVAC. Nello stesso periodo, veniva realizzato un computer con programma memorizzato chiamato EDSAC, presso l'Università di Cambridge in Inghilterra, sotto la direzione del Professor Maurice Wilkes. Con queste macchine e altre simili si entrò nell'era moderna dei computer. Sebbene fossero molto più lente, ingombranti e meno potenti dei sistemi attuali, EDVAC e EDSAC eseguivano i programmi in modo assai simile a quello dei computer miniaturizzati e infinitamente più potenti del ventunesimo secolo. Un modello commerciale di EDVAC, chiamato UNIVAC I (il primo computer effettivamente messo in vendita) fu realizzato da Eckert e Mauchly e acquistato dall'ufficio del censimento degli Stati Uniti il 31 marzo 1951 (funzionò per 12 anni prima di essere spento per l'ultima volta, smantellato e trasferito allo Smithsonian Institution). Questa data segna l'inizio effettivo dell'era informatica.

Il contributo di Von Neumann allo sviluppo dei sistemi informatici fu assolutamente fondamentale. Nonostante le sue proposte originali siano di circa 60 fa, praticamente ogni computer costruito oggi è una macchina di Von Neumann nel suo progetto di base.

John Von Neumann (1903–1957)

John Von Neumann nacque a Budapest, in Ungheria. Era un bambino prodigio in grado a sei anni di calcolare a mente divisioni di numeri a 8 cifre. Si rivelò un genio in tutti i suoi campi di studio, dalla fisica all'economia, all'ingegneria e alla matematica. A 18 anni ricevette il premio di miglior matematico dell'Ungheria, un Paese noto per l'alto livello nel campo. Emigrò negli Stati Uniti nel 1930, dove insegnò per tre anni all'Università di Princeton. Poi, nel 1933 divenne uno dei membri fondatori (insieme ad Albert Einstein) dell'Institute for Advanced Studies, dove lavorò per 20 anni.

Fu un brillante teorico che portò contributi pionieristici in matematica pura, ricerca operativa, teoria dei giochi e fisica teorica. Fu anche un ingegnere, interessato ad aspetti pratici e applicazioni del mondo reale, e questo lo portò a progettare e costruire il primo computer con programma in memoria. Uno dei primi computer costruiti da RAND Corp. nel 1953 fu chiamato "Johnniac" in suo onore, anche se Von Neumann lo detestava. Come UNIVAC I, ha un posto d'onore allo Smithsonian Institute.

Molto è cambiato in informatica, e una potente stazione di lavoro con grafica ad alta risoluzione sembra avere poco in comune con un EDVAC. Tuttavia, i principi elementari sui quali si basano questi sistemi sono praticamente identici e il loro funzionamento è soggetto al medesimo modello teorico. Un vecchio detto nel mondo dell'informatica sostiene che: "Non c'è nulla di nuovo dopo Von Neumann!". Questo non è ovviamente vero (molto in effetti è successo), ma dimostra l'importanza e la solidità concettuale del progetto originale di Von Neumann.

1.4.3 L'era moderna: dal 1950 a oggi

Gli ultimi 50 anni relativi allo sviluppo dei computer sono stati dedicati al miglioramento in termini di hardware e software dell'architettura di Von Neumann. Dal 1950 lo sviluppo dei sistemi informatici è stato principalmente un processo evolutivo e non rivoluzionario. Le numerose modifiche apportate ai computer nel corso dell'ultimo mezzo secolo li hanno resi più veloci, più piccoli, meno costosi, più affidabili e più facili da usare, ma senza cambiarne drasticamente la struttura di base sottostante.

Il periodo 1950–1957 (queste date sono solo approssimazioni) viene spesso definito come la **prima generazione** dell'informatica. Questa era vide l'apparizione di UNIVAC I, il primo computer commerciale, e di IBM 701, il primo computer creato dalla società che sarebbe presto divenuta leader in questo nuovo settore. Questi primi sistemi erano simili nel progetto a EDVAC ed erano ingombranti, costosi, lenti e inaffidabili. Utilizzavano le valvole per l'elaborazione e l'archiviazione dei dati e richiedevano una manutenzione assai complessa. La sola semplice operazione di accensione della macchina poteva bruciare una decina di valvole! Per questo motivo, le macchine della prima generazione venivano utilizzate solo da personale qualificato e solo in luoghi speciali, quali aziende di grandi dimensioni, laboratori di ricerca governativi e universitari e installazioni militari, in grado di fornire questo dispendioso ambiente di supporto.

La **seconda generazionee** dell'informatica, all'incirca nel periodo 1957–1965, preannunciò un importante cambiamento nella dimensione e complessità dei computer. Verso la fine degli anni '50, le ingombranti valvole furono sostituite con un singolo transistor con dimensione di pochi millimetri e la memoria fu realizzata mediante minuscoli nuclei magnetici con un diametro di 2 soli millimetri (entrambi questi dispositivi sono illustrati nel Capitolo 4). Queste tecnologie non solo ridussero notevolmente la dimensione dei computer, ma ne aumentarono anche l'affidabilità, riducendone il costo. Improvvisamente, l'acquisto e l'uso di un computer divenne una reale possibilità per le piccole e medie imprese, gli istituti scolastici e gli enti governativi. Questa fu inoltre l'era della comparsa di FORTRAN e COBOL, i primi **linguaggi di programmazione di alto livello** (simili al linguaggio naturale). Questo tipo di linguaggio di programmazione è illustrato nel Capitolo 8. Non era più necessario essere un tecnico elettronico per risolvere un problema al computer. Bastava sapere come scrivere i comandi in un linguaggio di alto livello. Era ufficialmente nata una nuova professione: quella di **programmatore**.

Questo processo di miniaturizzazione proseguì fino a giungere alla **terza generazione** dell'informatica, che durò all'incirca dal 1965 al 1975. Questa fu l'era dei **circuiti integrati**. Invece di utilizzare componenti elettronici discreti, transistor, resistenze e condensatori venivano incisi fotograficamente su un pezzo di silicio, il che riduceva ulteriormente la dimensione e il costo dei computer. Dalla dimensione che richiedeva

un intero edificio, e poi a quella che occupava una stanza, i computer si erano ora ridotti a una dimensione simile a quella di una scrivania, e questo periodo vide la nascita del primo **minicomputer**: il PDP-1 prodotto da Digital Equipment Corp. Vide inoltre la nascita dell'**industria del software**, con la comparsa dei produttori di programmi, come i pacchetti di contabilità e i programmi statistici, destinati a un numero sempre crescente di utenti di computer. Verso la metà degli anni '70, i computer non erano più una rarità; erano ormai ampiamente utilizzati in tutti i settori, da quello governativo, a quello delle forze armate, a quello dell'istruzione.

La **quarta generazione**, 1975–1985, vide la comparsa del primo **microcomputer**. La tecnologia dei circuiti integrati era avanzata a tal punto che un intero sistema di computer poteva essere contenuto su una singola piastra di circuiti che poteva stare in una mano. I sistemi che nei primi anni '70 erano grandi come una scrivania, ora potevano essere collocati sopra la scrivania stessa, essendo grandi poco più di una macchina per scrivere. Nella Figura 1.7 è illustrato l'Altair 8800, il primo microcomputer al mondo, apparso nel gennaio del 1975.

Presto divenne inusuale la mancanza di un computer sulla scrivania di qualcuno. Il settore del software forniva diversi tipi di pacchetti innovativi, fogli elettronici, database e programmi di grafica, al fine di soddisfare l'esigente popolazione degli utenti. Questa era vide la comparsa delle primi **reti di computer**, poiché gli utenti si resero conto che buona parte della potenza dei computer risiede nella loro capacità di agevolare le comunicazioni con gli altri (le reti sono illustrate in dettaglio nel Capitolo 7.) Un'applicazione importante fu la **posta elettronica**. Poiché moltissimi utenti di computer erano principianti e inesperti, emerse il concetto di sistemi **user-friendly** (semplici e amichevoli). Ciò comprendeva nuove interfacce utente grafiche con menu a discesa, icone e altri aiuti visuali per rendere più facile e divertente l'esperienza con il computer. I **sistemi embedded**, dispositivi che contengono un computer per controllarne il funzionamento interno, apparvero per la prima volta durante questa generazione. I computer stavano diventando sufficientemente piccoli da poter essere collocati all'interno delle auto, nei termostati, nei forni a microonde e negli orologi da polso.

Il computer alla televisione

Agli inizi dell'era dei computer (1951–1952), poche persone sapevano che cosa fosse un computer e ancora meno ne avevano visto o utilizzato uno.

I computer erano utilizzati soltanto da un piccolo gruppo di specialisti di alto livello con alle spalle studi in matematica, fisica e ingegneria. In quegli anni, la conoscenza dell'informatica si limitava ai robot e ai computer alieni dei film di fantascienza.

Tutto cambiò nel novembre del 1952, quando milioni di americani accesero la televisione per vedere i risultati della sfida presidenziale tra Dwight D. Eisenhower e Adlai Stevenson. Oltre a tanti giornalisti, il pubblico vide un nuovo membro dello staff: un UNIVAC I. La CBS aveva noleggiato un computer, installandolo al centro della scena con tutte le luci e i nastri in azione, pensando di usarlo per prevedere i risultati delle elezioni, battendo sul tempo la concorrenza. Per colmo di ironia, l'UNIVAC predisse, sulla base di tecniche di campionamento statistico, che le elezioni sarebbero state vinte da Eisenhower, ma i dirigenti della CBS erano talmente scettici su quella nuova tecnologia che non pubblicarono la notizia finché non fu confermata anche dai metodi manuali.

Fu la prima volta che milioni di telespettatori poterono vedere un "computer digitale elettronico". Il personale della CBS lo trattava come se fosse umano. Il successo fu talmente grande che i computer furono poi usati spesso nei primi anni della TV, soprattutto nei giochi a quiz, dove promosse nel pubblico il concetto di "cervello elettronico".

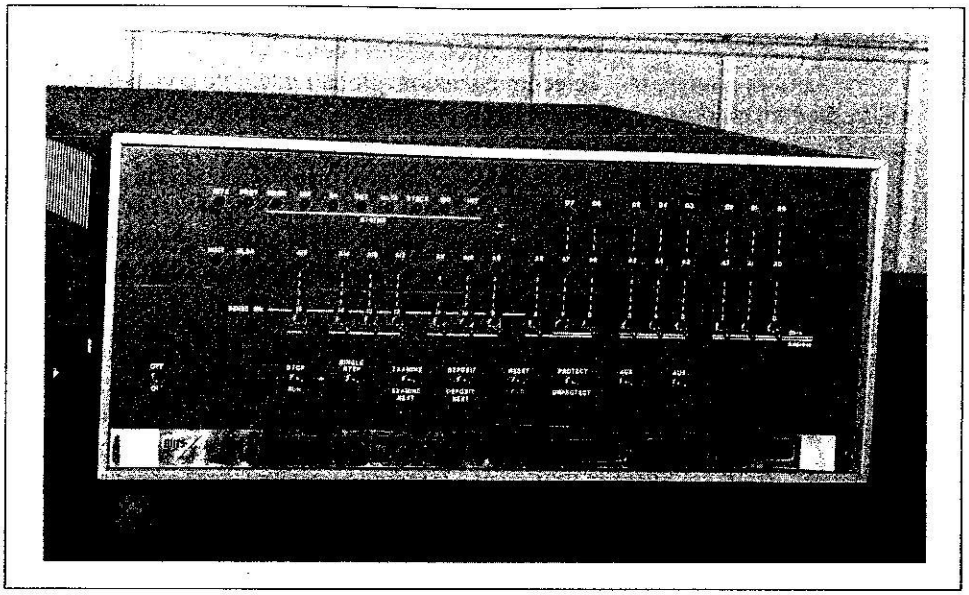


Figura 1.7
L'Altair 8800, il primo microcomputer del mondo.

La **quinta generazione**, 1985–?, è quella in ci troviamo oggi. Tuttavia, molto sta cambiando, e così rapidamente che la maggior parte degli scienziati ritiene che il concetto di generazioni distinte abbia esaurito la propria utilità. In informatica, il cambiamento è ora un elemento costante.

Tra i recenti sviluppi in informatica vi sono i seguenti.

- Imponenti processori in parallelo capaci di eseguire trilioni di operazioni al secondo.
- Dispositivi palmari e altri tipi di PDA (Personal Digital Assistant).
- Grafica ad alta risoluzione per elaborazione di immagini, creazione di filmati e realtà virtuale.

Il primo microcomputer del mondo

L'Altair 8800, mostrato nella Figura 1.7, fu il primo microcomputer e fece il suo debutto sulla copertina di *Popular Electronics* nel gennaio del 1975. Il suo sviluppatore, Ed Roberts, era il proprietario di un piccolo negozio di elettronica ad Albuquerque, New Mexico. La sua azienda era in cattive acque quando lesse di un nuovo microprocessore, l'Intel 8080. Roberts pensò che quel nuovo chip potesse essere usato per vendere un personal computer completo in forma di kit. Acquistò i chip da Intel al prezzo di 75 dollari l'uno e li usò per realizzare un kit denominato Altair 8800 (dal nome di un sito citato nella serie TV Star Trek), che commercializzò al prezzo di 397 dollari. Roberts pensava di poter vendere qualche centinaio di kit l'anno, e arrivò a venderne migliaia al giorno!

L'Altair era difficile da assemblare e aveva solo 256 celle di memoria, nessun dispositivo di ingresso/uscita e nessun supporto per il software. Per programmarlo, l'utente doveva inserire istruzioni in linguaggio macchina direttamente dagli interruttori della console. Ma anche così le persone lo amavano, perché era un computer vero.

Il chip Intel 8080 aveva la capacità di eseguire programmi scritti nel linguaggio chiamato BASIC che era stato sviluppato a Dartmouth nei primi anni '60. Una piccola azienda di software dello stato di Washington scrisse a Ed Roberts una lettera per informarlo che possedeva un compilatore BASIC che poteva funzionare sul suo Altair, rendendolo molto più facile da usare. L'azienda si chiamava Microsoft, e il resto è storia.

- Potenti interfacce utente con suoni, riconoscimento vocale, comandi tattili, fotografia, video e televisione.
- Telecomunicazioni globali integrate con dati, televisione, telefono, fax, Internet e il World Wide Web.
- Comunicazioni dati wireless.
- Imponenti dispositivi di memorizzazione in grado di contenere centinaia di trilioni di dati.
- Informatica pervasiva, nella quale i computer sono incorporati nelle auto, negli elettrodomestici della cucina, nei sistemi di riscaldamento domestici e persino nei vestiti.

In soli 50–60 anni, i computer sono progrediti dall'UNIVAC I, che costava milioni di dollari, aveva poche migliaia di locazioni di memoria ed era in grado di eseguire solo poche migliaia di operazioni al secondo, all'odierna stazione di lavoro al vertice della gamma, con monitor piatto ad alta risoluzione, miliardi di celle di memoria, quantità imponente di memoria esterna e abbastanza potenza di calcolo da eseguire più di 1 miliardo di istruzioni al secondo, per meno di mille euro. Cambiamenti di questa entità non si sono mai verificati in alcun altro settore tecnologico. Se lo stesso tasso di cambiamento si fosse verificato nel settore delle automobili, a partire dal Modello T del 1909, le auto odierne sarebbero in grado di viaggiare a una velocità superiore a 10.000 km all'ora, farebbero circa mezzo milione di km con un litro e costerebbero soltanto 1 euro!

Nella Figura 1.8 è riportato un riepilogo dei principali sviluppi che si sono verificati durante ciascuna delle cinque generazioni di sviluppo dei computer. E alla base di tutti questi incredibili miglioramenti, il modello teorico che descrive il progetto e la costruzione dei computer non è cambiato nell'arco degli ultimi 50 anni.

Tuttavia, molte persone sono convinte che siano in procinto di apparire significativi e importanti cambiamenti strutturali. Alla fine del Capitolo 5 verranno introdotti i modelli di calcolo che sono fondamentalmente assai diversi dall'architettura di Von Neumann in uso oggi. Queste soluzioni del tutto nuove (per esempio l'informatica quantistica) potrebbero essere modelli utilizzati nel ventiduesimo secolo e oltre.

1.5 Struttura del testo

Questo libro è suddiviso in sei sezioni separate, dette **livelli**, ognuna delle quali è dedicata a un aspetto della definizione di informatica riportata all'inizio di questo capitolo. Ora ripetiamo tale definizione e vediamo come corrisponde alla sequenza di argomenti presentati nel libro.

Definizione

L'**informatica** è lo studio degli algoritmi, che comprende:

1. le loro proprietà formali e matematiche;
2. le loro implementazioni hardware;
3. le loro implementazioni linguistiche;
4. le loro applicazioni.

Figura 1.8
Cronologia di alcuni tra i principali progressi compiuti in informatica.

Generazione	Date approssimate	Progressi principali
Prima	1950–1957	Primi computer commerciali. Primi linguaggi di programmazione simbolici. Uso di aritmetica binaria, valvole per memorizzazione. Ingresso/uscita su schede perforate
Seconda	1957–1965	Transistor e memorie centrali. Primi dischi per memoria di massa. Dimensioni inferiori, maggiore affidabilità, costi più bassi. Primi linguaggi di programmazione di alto livello. Primi sistemi operativi.
Terza	1965–1975	Circuiti integrati. Ulteriore riduzione in dimensioni e costi, più affidabilità. Primi minicomputer. Sistemi operativi in time-sharing. Comparsa dell'industria del software. Prima serie di standard per compatibilità tra sistemi.
Quarta	1975–1985	Circuiti integrati LSI e VLSI. Ulteriore riduzione in dimensioni e costi, più affidabilità. Primi microcomputer. Crescita dell'industria del software e sviluppo di nuovi tipi di software. Reti di computer. Interfacce utente grafiche.
Quinta	1985–?	Circuiti integrati ULSI. Supercomputer e processori paralleli. Computer portatili e palmari. Reti wireless. Dispositivi con enorme capacità di memoria di massa. Informatica pervasiva. Grafica ad alta risoluzione, tecnologie di visualizzazione, realtà virtuale. Reti mondiali. Interfacce utente multimediali. Diffusione dell'uso di suoni, immagini e filmati digitalizzati.

L'informatica è lo studio degli algoritmi, che comprende

1. *Le loro proprietà formali e matematiche.* Il Livello 1 del libro (Capitoli 2 e 3), intitolato "Fondamenti algoritmici dell'informatica", continua la discussione della risoluzione algoritmica dei problemi iniziata nei Paragrafi 1.2 e 1.3 di questo capitolo, introducendo importanti proprietà matematiche e logiche degli algoritmi. Il Capitolo 2 presenta lo sviluppo di numerosi algoritmi che risolvono importanti problemi tecnici (molto più "tecnici" del lavaggio dei capelli); inoltre esamina concetti legati al processo di risoluzione dei problemi, come il modo in cui possiamo scoprire e creare buoni algoritmi, la notazione che possiamo utilizzare per esprimere le nostre soluzioni e i metodi per verificare se l'algoritmo proposto risolve correttamente il problema in esame.

L'esempio del computer che gioca a scacchi con la "forza bruta" mostra che non basta sviluppare un algoritmo corretto: occorre anche trovare una procedura di risoluzione efficiente e che produca il risultato desiderato in un tempo ragionevole (mettereste sul

mercato un programma per gli scacchi che richieda 10^{48} anni per fare la prima mossa?). Il Capitolo 3 spiega come confrontare l'efficienza di diversi algoritmi e scegliere quello più adatto per risolvere un problema dato. I materiali forniti nel Livello 1 offrono i necessari fondamenti per lo studio dell'informatica.

2. *Le loro implementazioni hardware.* Nella nostra introduzione all'informatica abbiamo esaminato come si comporta un algoritmo quando è eseguito da un "agente di calcolo" astratto, ma nella realtà vogliamo eseguire gli algoritmi su macchine "reali" per ottenere risposte "reali". Il Livello 2 del libro (Capitoli 4 e 5) è intitolato "Hardware" e spiega come progettare e costruire computer. L'argomento è considerato da diversi punti di vista. Il Capitolo 4 presenta una dettagliata discussione sull'hardware di base, presentando gli elementi fondamentali dei computer (numeri binari, transistor, porte logiche, circuiti) e mostrando come dei dispositivi elettronici elementari possano essere utilizzati per costruire componenti in grado di eseguire operazioni aritmetiche e logiche come addizione, sottrazione, confronto e disposizione in sequenza.

Benché sia interessante e importante, questa prospettiva produce una vista di basso livello del computer. È difficile capire come funziona un computer studiando soltanto questi componenti elementari, così come sarebbe difficile capire il comportamento umano studiando le singole cellule. Perciò il Capitolo 5 presenta una vista dell'hardware di livello superiore: esamina i computer non come ammassi di fili e circuiti, ma come insieme integrato di sottosistemi denominati memoria, processore, memoria di massa, input/output e comunicazioni. In questo capitolo vengono spiegati in dettaglio i principi dell'architettura di Von Neumann introdotti nel Paragrafo 1.4 di questo capitolo.

Lo studio dei computer può essere svolto a un livello ancora più alto. Per capire come funziona il computer, non occorre esaminare il funzionamento di ciascuna delle numerosissime componenti interne alla macchina; è sufficiente comprendere bene pochi elementi critici che sono indispensabili per il nostro lavoro. Dal punto di vista dell'utente, tutto il resto è superfluo. Questa vista del computer e delle sue risorse "orientata all'utente" è detta **macchina virtuale** o **ambiente virtuale**. Una macchina virtuale è composta soltanto delle risorse che l'utente percepisce, e non di tutte le risorse hardware realmente esistenti. Questo punto di vista è analogo al nostro livello di comprensione di ciò che accade sotto il cofano di un'automobile. Possono esserci migliaia di componenti meccanici all'interno del motore, ma la maggior parte di noi si preoccupa soltanto degli elementi riportati sul cruscotto: pressione dell'olio, livello del carburante, temperatura del motore. Questo è il nostro "motore virtuale", tutto ciò che ci serve o vogliamo conoscere. Siamo tutti felicissimi di lasciare i dettagli del motore al nostro caro amico meccanico.

Il Livello 3 (Capitoli 6 e 7), intitolato "Macchina virtuale", descrive come si crei un ambiente virtuale utilizzando un componente denominato **software di sistema**.

Il Capitolo 6 esamina il più importante e comune software di sistema per i moderni computer, il **sistema operativo**, che controlla l'attività complessiva del computer e facilita l'accesso agli utenti. Il Capitolo 7 descrive come questo ambiente virtuale possa oltrepassare i limiti di un singolo sistema e spiega come interconnettere singole macchine in **reti di computer** e **sistemi distribuiti** che consentono agli utenti di accedere a un'enorme quantità di computer e informazioni, oltre che di altri utenti. È il software di sistema, con la macchina virtuale da esso creata, che rende gestibile e utilizzabile l'hardware del computer.

3. *Le loro implementazioni linguistiche.* Dopo aver studiato la struttura dell'hardware, l'organizzazione del computer e le macchine virtuali, conoscerete bene le tecniche utilizzate per progettare e costruire computer. Come si fa a utilizzare l'hardware per risolvere problemi importanti e interessanti? Il Livello 4, intitolato "Software" (Capitoli 8–10) esamina la progettazione e l'implementazione di software per computer, considerando i programmi e le sequenze di istruzioni eseguite dall'hardware, invece dell'hardware in sé.

Il Capitolo 8 presenta alcuni concetti fondamentali legati alla programmazione di computer. Questo singolo capitolo non ha certo l'ambizione di trasformare il lettore in un esperto programmatore: vuole semplicemente illustrare alcune caratteristiche fondamentali dei moderni linguaggi di programmazione e fornire un'idea di che cosa sia la programmazione e il lavoro del programmatore.

Il Capitolo 9 spiega come un programma scritto in un linguaggio di alto livello possa essere tradotto nei codici del linguaggio macchina di basso livello descritti nel Capitolo 5. Infine, il Capitolo 10 mostra che, perfino con tutto l'hardware e il software descritti nei primi 9 capitoli, esistono problemi che non possono essere risolti mediante algoritmi, dimostrando così che l'informatica ha dei limiti.

4. *Le loro applicazioni.* La maggior parte delle persone non si preoccupa di creare programmi, ma di utilizzarli, un po' come avviene per le automobili: ci sono pochi ingegneri meccanici, ma molti guidatori. Il Livello 5, intitolato "Applicazioni" (Capitoli 11–12), si occupa non di *come* scrivere un programma, ma di *che cosa* possono fare i programmi.

I Capitoli 11 e 12 esaminano soltanto alcune delle sempre più numerose applicazioni dei computer, come l'intelligenza artificiale, la visualizzazione, la grafica, la crittografia, la simulazione e il commercio elettronico. Naturalmente non è possibile descrivere tutti i modi in cui i computer sono utilizzati oggi o lo saranno nel futuro. In ogni caso, è difficile trovare un'area della nostra società moderna e complessa che non sia influenzata in qualche modo dalle tecnologie dell'informazione. I lettori interessati ad applicazioni non discusse qui possono consultare la bibliografia di riferimento indicata per approfondire la propria area di interesse.

Alcuni professionisti dell'informatica non si occupano di costruire computer, creare programmi o utilizzare le applicazioni appena descritte: si interessano dell'impatto sociale e culturale, con aspetti positivi e negativi, di questa tecnologia in eterna evoluzione. Il Livello 6, intitolato "Aspetti sociali" (Capitolo 13), si occupa di questo campo, che non rientra nella definizione originale di informatica, ma ha acquisito grande importanza. In questo livello passiamo al massimo livello di astrazione, allontanandoci sempre più dal computer, per discutere aspetti sociali, etici, legali e professionali legati al computer e alla tecnologia dell'informazione. Questi aspetti sono molto importanti, perché anche le persone non direttamente coinvolte nello sviluppo o nell'uso dei computer sono profondamente influenzate da essi, esattamente come la società è stata alterata in maniera drastica e permanente da tecnologie come il telefono, la televisione e l'automobile.

Quest'ultimo capitolo esamina complessi argomenti come i crimini informatici, la riservatezza dei dati, la proprietà intellettuale. Poiché è impossibile risolvere le complesse questioni che nascono in questi campi, l'intento è semplicemente quello di fornire al lettore maggiore consapevolezza e offrire alcuni strumenti decisionali per aiutarlo a trarre le proprie conclusioni.

La gerarchia complessiva a sei livelli è illustrata nella Figura 1.9. Tale struttura organizzativa è uno dei più importanti aspetti di questo libro. Per descrivere un campo di studio, non è sufficiente presentare una massa di fatti e spiegazioni. Perché il lettore possa assorbire, comprendere e integrare tali informazioni occorre fornire un tema, una relazione, un percorso che colleghi le varie parti della narrazione: in sostanza, un "quadro generale". Il nostro quadro generale è quello della Figura 1.9.

Iniziamo con i fondamenti dell'informatica (Livello 1) e procediamo verso l'alto attraverso cinque distinti livelli di astrazione, dai dettagli di basso livello come i circuiti elettronici e l'hardware (Livello 2), attraverso livelli intermedi che trattano le macchine virtuali (Livello 3), i linguaggi di programmazione e lo sviluppo di software (Livello 4), fino a livelli più elevati che studiano le applicazioni dei computer (Livello 5), l'uso e l'impatto delle tecnologie dell'informazione (Livello 6). Il materiale di ciascun livello fornisce le basi per mettere in luce la bellezza e la complessità di una visione più astratta dell'informatica.

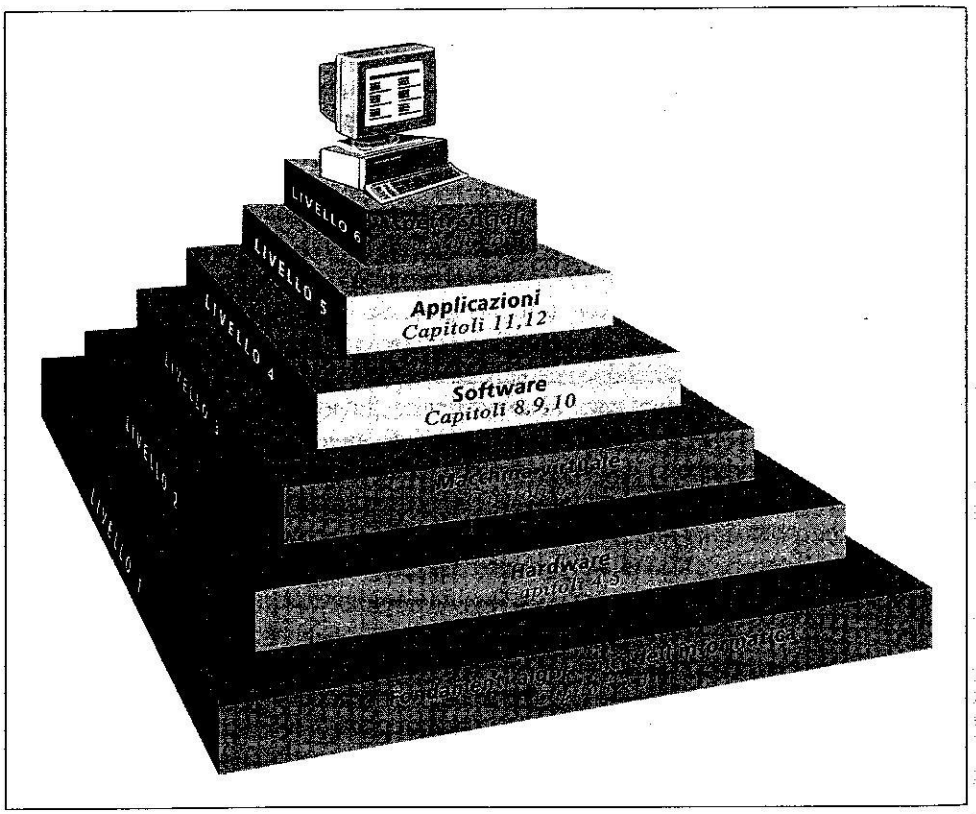


Figura 1.9
Organizzazione del testo in una gerarchia a sei livelli.

Letture di approfondimento

I testi elencati di seguito forniscono una buona panoramica sull'informatica. Come questo stesso libro, essi trattano molti aspetti della disciplina.

Biermann, A. W. *Great Ideas in Computer Science*, 2nd ed. Cambridge, MA: MIT Press, 1997; tr. it. *Informatica. Una panoramica generale*, Pearson Education Italia, 2004.

Brookshear, J. G. *Computer Science: An Overview*, 8th ed. Reading, MA: Addison Wesley, 2005; tr. it. *Informatica. Una panoramica generale*, Pearson Education Italia, 2004.

Decker, R., e Hirshfield, S. *The Analytical Engine: An Introduction to Computer Science Using the Internet*, Boston, MA: Course Technology, 2004.

Dewdney, A. K. *The New Turing Omnibus*. New York: Freeman, 2001.

Dewdney, A. K. *Introductory Computer Science: Bits of Theory, Bytes of Practice*. Boston, MA: W.H. Freeman & Company, 1996.

I testi che seguono forniscono un'eccellente panoramica sull'evoluzione storica dei computer e del software.

Broy, M., e Denert, E. *Software Pioneers*. Amsterdam: Springer-Verlag, 2002.

Cambell-Kelly, M., e Asprey, W. *Computers: A History of the Information Machine*. New York: Basic Books, 1997.

Ceruzzi, P. A *History of Modern Computing*. 2nd Edition, Cambridge, MA: MIT Press, 2003; tr. it. *Storia dell'informatica. Dai primi computer digitali all'era di Internet*, Milano, Apogeo, 2006.

Ifrah, George. *The Universal History of Computing: From the Abacus to Quantum Computer*. New York: Wiley, 2002.

Rojas, Ral, Hashagen, Ulf, Rojas, Raul, *The First Computers—Their History and Architecture*, Cambridge, MA: MIT Press, 2002.

Wurster, C. *The Computer: An Illustrated History*. Cologne, Germany: Taschen, 2002

Inoltre, un'ottima risorsa per informazioni sulla storia delle tecnologie dell'informazioni e del relativo impatto sulla società è fornita dal Charles Babbage Institute dell'Università del Minnesota; il sito Web è www.cbi.umn.edu.

Esercizi

1. Identificate degli algoritmi, diversi da quelli del videoregistratore e delle ricette di cucina mostrati nel testo, che incontrate nella vita quotidiana. Scriveteli in una notazione adatta e spiegate perché soddisfano tutti i criteri distintivi degli algoritmi presentati nel capitolo.
2. Nelle istruzioni per il videoregistratore della Figura 1.1 il passo 4 dice "Inserisci il numero del canale che vuoi registrare e premi il pulsante CANALE". Si tratta di un'operazione non ambigua e ben definita? Spiegate perché sì o perché no.
3. Tracciate l'esecuzione dell'algoritmo di addizione decimale della Figura 1.2 utilizzando i seguenti valori di input:
   $m = 3$ $a_2 = 1$ $a_1 = 4$ $a_0 = 9$
   $b_2 = 0$ $b_1 = 2$ $b_0 = 9$
   A ogni passaggio mostrate i valori per c_3, c_2, c_1, c_0 e *riporto*.
4. Modificate l'algoritmo di addizione decimale della Figura 1.2 in modo che non stampi gli zeri non significativi in testa. In pratica, la risposta alla domanda 3 deve apparire come 178 anziché 0178.
5. Sotto quali condizioni la nota formula quadratica:
   $$Radici = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$
   non è effettivamente computabile? (supponete di lavorare con numeri reali).
6. Confrontate le due soluzioni per l'algoritmo dello shampoo mostrato nelle Figura 1.3(a) e 1.3(b). Quale pensate che sia una soluzione più generale? Perché? (*suggerimento*: e se voleste lavarvi i capelli 1000 volte?).